**Scenario 1: Creating a new project workflow:**

Create an empty remote repo on github called “demo1”

clone remote repo

edit a readme.txt and check in (using gitext)

Edit readme.txt and checkin (using git bash)

Show commits

Show GitExt of checkins.

push to remote

view on github.

**Scenario 2: Using Git to work offline for existing subversion project**

Use GitExt to “create new repo” for C:\TEMP\GIT-demo\jboss-modules

Modify a readme.txt and commit.

Modify a readme.txt and commit.

Show collapse of commits by checkout of old commit 2 commits ago, then force commit to undo the 2 newer commits.

Create a branch. Update pom.xml. merge into master.

**Scenario 3: Show usage of remote branch for reset.**

Checkout demo1 project

Got to master branch.

Create a commit and see the difference between local and remote branch.

Make bad commits, then can revert back to remote master using reset.

This shows there is always a “backup good” version in remote.

**Scenario 4: Show stash push/pop (instead of making a copy of backup)**

Check out master

Create a new file test3.java

A new bug comes and need to get fix right away. Don’t want to check in test3.java since it’s not ready yet.

Stash test3.java

Now work on the bug fix. Update README.txt and checkin to master.

Pop test3.java and continue working.

**Scenario 5: Show merge and rebase**

Checkout demo1 project

Create branch “branch1”

Add a test.java with a commits to local branch1.

Update test.java with another commit to local branch1.

Checkout local master and merge branch1 using “always create new merge commit”. Show the loop.

Create branch “branch2”

Add a test2.java with a commits to local branch2.

Update test2.java with another commit to local branch2.

Checkout master

Rebase branch2 onto master. Show “straight line”. Rebase means each commit of branch2 is now commits of the master. In merge, the branch1 commits is separate from master.

Clone the demo1 repo another another directory. Update readme.txt and push to master.

On 1st repo, do a pull and merge master to branch. Show the loop.

Undo the merge and show a rebase master to branch and show the “straight line”. Undo the rebase.

Check out master and merge branch to master.

How to create new repo and checkin local.

https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/

1. In github create a new repo. Don’t add gitignore or README. Make note of the remote URL
2. Create you local remote. “git init”
3. “git add .”
4. git commit –m “initial”
5. git remote add origin remote-url-in-step1 (origin could be anything, it could be main)
6. git remote –v to verify

origin https://github.com/h-t-tran/gitdemo.git (fetch)

origin https://github.com/h-t-tran/gitdemo.git (push)

1. git push origin master (origin could be main)

OR,

You can d+o “git clone remote-url”, then you can just do “git push”

Advantage of distributed SCM

1. Hierarchial – developers push(checkin) to a subsystem level repo. Subsystem level pushed to a main repo
2. Distributed - Developer push the code to his/her own repo. Integrator pick and choose and pull (checkout) developers’ repos into an official repo.
3. Developers can work offline in a disconnected network using local repo. Push changes to main repo when connected.
4. Easy to backup – just clone a repo into a DVD.
5. Support centralized SCM – developers push to a single repo.

GIT started in 2005.

**Install GIT on Windows**

msysgit project. 2.14.1 is most recent.

Install in C:\Program Files\Git2.13.3\bin, run bash.exe to bring up git shell.

“git –version”

<https://git-for-windows.github.io/>

**Install GIT on Linux**

sudo apt-get install git-core

**Configuration**

git config --global –list, but easier to do configuration in gitextension.

Git config --global color.ui auto

**Create local repo**

Git init

Git status

Git add (add new file to staging, must add before commit)

Git add –u (only add updated already checked in files)

Git commit (move staging to local repo)

Git log // to display the log and version and hash.

log --graph // display tree view

git diff firsthash 2ndhash

git diff HEAD~2 HEAD~1 // compare 1st prev version with 2nd prev version

git checkout file.txt // to undo any changes. Same as reset to HEAD

git reset --hard // undo all

.gitignore (such as log or bin. Eg logs/ means ignore all logs dir, or /dir1/logs means abs path)

**WORKING W/ REMOTE**

Git clone <https://github.com/jquery/jquery.git>

Git log –graph to view tree

Fetch, merge

Pull = fetch + merge

Git show HEAD~1

Git tag // show all stable points.

Git tag mytag // tag the current commit with name “mytag” locally

Git push --tag // push the tags to remote repo

If you have a local repo and not yet check to remote

Git remote –v // show the remote version. If empty, there is not remote repo so other ppl won’t see it.

**Branches**

**Stashing**

* Use to save uncompleted work while a critical fix is needed from production
* Use stash instead of zip up code for backup.

Git stash

Git stash list

Git stash apply or git stash pop // pull the stash code back into the branch. Pop removes the stash from the stack. Where apply keeps it in stack.

Git stash drop // to delete the stash head.

**Merge**

Kdiff 3 way merge

Base = the “original” copy where the local and remote copies came from.

Local = your local copy in local master.

Remote = remote copy you want to merge to.

To show conflict, create a branch1, mod a file at the same line. Create branch2, mod same file at same line. Then merge branch2 to master. Now back to branch 1 and try to merge to master. This should result in conflict.

**Rebase**

Taking a current master commit and apply it to current branch which will move (replay) the current branch commits on top of master.

Use rebase when don’t want intertwine branches.

Have better git history**.**

**Cherry Pick**

Use when you want just a single commit from another branch into master/your-branch and don’t want to merge or rebase from that branch which brings in all other fixes.

Use to apply a single patch to a master.

**Delete remote branch**

git push origin --delete branch1

DEMO Plan

1. **Create local repo.**

* git init

show .git repo

create README.txt

* git status 🡪 show untrack files.
* git add README.txt 🡪 add to staging.
* git commit –m “initial checkin” 🡪 check in to local repo. The file is track by git

modify README.txt

* git status
* git commit –m “second checkin”
* Open Git Extension to view the repo.

1. **Show loading of current files.**

How does local repo help us?

Use git to track C:\TEMP\GIT-demo\jboss-modules SVN repo when offline?

Use GitExtension to track jboss-modules.

In GitExtension, add .svn/\* to git ignore by select files and right click and Add File to Gitignore. Show .gitignore file

Add other files to git Extension and commit.

Modify jboss-module/pom.xml and show changes. Bring up git bash (git status). Use GitExtension to checkin.

Now we have a version ctrl w/out needing to connect to version ctrl server.

When go online, then can commit to SVN.

1. **Show create a new repo on github (or on-premise git server).**

Clone

Add files

Commit

push

1. **Show collapse commits.**

Why do we want to do that?

1. **How to pull and resolve conflict.**

Make 2 clone repos another dir1, dir2

From dir2, mod README.txt and push

From dir1, mod README.txt same file and commit to local. Now pull and resolve conflict. Show kdiff. Resolve and commit.

1. **Create branch and Merge.**

Use gitExt to create branch1, create a new file and commit to branch1. Push branch to remote.

1. **Delete branch**

Delete local branch and change mind. Not total lost, the data still available on remote and can recreate local branch.

Delete remote branch “git push origin --delete branch1”

1. **Stash**

Push, pop.